International Journal of Mathematics and Computer Applications Research (IJMCAR)
ISSN(P): 2249-6955; ISSN(E): 2249- 8060
Vol. 5, Issue 5, Oct 2015, 65-72

LEARNING RATE COMPUTATION FOR THE BACK PROPAGATIONALGORITHM

ABBAS H. TAQI

Department of Mathematics, College of Science, University of Kirkuk, Iraq

ABSTRACT

The classical back propagation (CBP) method is the simplest algorithm for training feed-forward neural networks (FFNNs). It uses the steepest descent search direction with fixed learning rate α to minimize the error function E, since α is fixed at each iteration this cause slow convergence. In this paper I will suggest a new formula for computing learning rate α by using tangent hyperbolic function with Wolfe conditions to accelerate the convergence of the CBP algorithm. Simulation results are presented and compared with matlab toolbox training algorithms.

KEYWORDS: MLFFs, Weight Efficient, Training Algorithm, Back Propagation, Learning Rate, Zoutendijt Theorem, FFNN, Spect

1. INTRODUCTION

The area of artificial neural networks has been extensively studied and has been widely used in many applications of artificial intelligence. Due to their excellent capability of self-learning and self-adapting, they attracted much interest from the scientific community. Recently, they have been established as vital components of many systems and are considered as a powerful tool for solving different type of problems [5], [19]. There exist many types of neural networks e. g see[7,12,17], but the basic principles are similar. Each neuron in the network is able to receive input signals, to process them and to send an output. Each is connected at least with one neuron, and each connection is evaluated by a real number, called weight coefficient that reflects the degree of importance of the given connection in the neural networks.

Multi-layer feed-forward neural networks (MLFFs) have been the preferred neural network architecture for the solution of classification, function approximation and other types of problems, due to their outstanding learning and generalization abilities. MLFFs allow signals to travel one way only, from input layer to the output layer. There is no feedback (loops) i.e the output of any layer does not affect that same layer. MLFFs tend to be straightforward networks that associated inputs with outputs [7, 3, 19]

There exist two main types of training process supervised and unsupervised training. Supervised training means that neural network, has known desired output (target) and adjusting of weight coefficients is done in such a way that the calculated and the desired outputs are as close as possible [6].This paper is concerned with supervised learning, for unsupervised learning see [9].

The remainder of this paper is organized as follows. Section 2 describes a brief summary of supervised training process with classical back propagation algorithm. Section 3 gives the suggested method and it's global convergence analysis. Section 4 contains the experimental results and finely section 5 presents the concluding remarks.

2. SUPERVISED TRAINING PROCESS AND CLASSICAL BACKPROPAGATION ALGORITHM

The training of multi-layer feed-forward consists of neurons that are ordered into layers. The first layer is called the input layer the last layer is called output layer and layers between them are hidden layers can be formulated as the minimization of an error function E(w) see equation(1) below that depends on the connection weights w of the network and defines as the sum of the squared differences between the computed and target values

$$E(w)=\frac{1}{2}\sum_{j=1}^{p}\sum_{i=1}^{M}(T_i^{(j)}-O_i^{(j)})^2 \tag{1}$$

The variables O_i and T_i stand actual and desired response of ith output neurons, respectively. The superscript j denote the particular learning pattern p. The vector W is composed of all weights in the net, summation of the actual errors takes place over all M output neurons and all P learning data (X, T) where the N-dimensional vector X is the input vector and the M-dimensional vector T is the target vector associated with X. The most popular training algorithm for MLFF neural network is the classical Back propagation [18] which may be proceed in one of two basic ways. Pattern mode (on-line) or batch mode. In pattern mode of CBP learning, weight updating is performed after the presentation of each training pattern. In the batch mode of CBP learning weight updating is performed after the presentation of all the training examples (i.e. after the whole epoch) see [4, 12]. This paper consider the batch mode training. The high level description of the classical back propagation training algorithm for MLFFs is

CBP Algorithm

Step (1): initiate k = 1,w_k, α = 0.01, tolerance E_{rr} and ε→0

Step (2): If $E(w_k) < E_{rr}$ or $\|\nabla E(w)\| < \varepsilon$ terminate

Step (3): compute descent search direction

$$d_k = -g_k \qquad \forall\ k$$

Step (4): update the weights

$$w_{k+1} = w_k + \alpha\ d_k$$

Step (5): set k=k+1 and go to Step (2).

Although the CBP algorithm is a simple learning algorithm for training, since only two major parameters are used to control the training process. The first is the learning rate α which is constant value, second is the search direction $d_k = -g_k \quad \forall\ k$, unfortunately it is not based on a sound theoretical basis and it has some disadvantages, first it's convergence is fast only if the parameter setting is nearly optimal, furthermore the convergence rate is slow and decreases rapidly as the problem size increase, second it's convergence is guaranteed only if the learning rate is small enough[2,10]. The mean problem then is to determined a priority what small enough means. In other words for shallow minimum, the learning rate is often too small where as for narrow minimum it is often too large and process never convergence, there for

the CBP tends to be inefficient. From the above discussion we see that the learning rate α is crucial for the convergence as well as to speed up of the algorithm. Many approaches have been proposed to compute learning rate, see [1, 11].

3 SUGGESTED METHOD AND It's CONVERGENCE ANALYSIS

3.1 Suggested Method

After d_k is fixed ($d_k = -g_k \quad \forall\ k$) the learning rate α ideally would solve the following one dimensional optimization problem

$$\underset{\alpha>0}{Min}\ E(w_k + \alpha\, d_k) \tag{2}$$

This optimization problem is usually impossible to solve exactly, instead α is computed (via an iterative procedure referred to as line search) either to approximately solve the above optimization problem or to ensure sufficient decrease in the value of E. In order to avoid solving equation (2) we need to find α such that the following Wolfe conditions hold

$$E(w_k + \alpha_k d_k) - E(w_k) \le c_1 \alpha_k g_k^T d_k \tag{3}$$

$$g(w_k + \alpha_k d_k)^T d_k \ge c_2 g_k^T d_k \tag{4}$$

Where $0 < c_1 < c_2 < 1$.

To find such α_k we shall develop a new procedure which contains two stages, the first stage computes, α_k from the Tangent hyperbolic equation

$$\alpha_{k+1} = \frac{1-e^{-\alpha_k}}{1+e^{-\alpha_k}} = 2*(\frac{1}{1+e^{-\alpha_k}}) - 1$$

$$\alpha_{k+1} = 2 * S(\alpha_k) - 1 \tag{5}$$

Where $S(\alpha)$ is the sigmoid transfer function used in the network. Second stage use the backtracking to satisfy the Wolfe conditions.

At this point, we present a high level description of proposed algorithm (ABBP say) for neural network training.

Algorithm (ABBP)

Step (1): initiate k=1, w_k, $0 < c_1 < c_2 \le 1$, , tolerance E_{rr} and $\varepsilon \to 0$

Step (2): If $E(w_k) < E_{rr}$ or $\|\nabla E(w)\| < \varepsilon$ terminate

Step (3): compute descent search direction

$$d_k = -g_k \qquad \forall\ k$$

Step (4): Calculate the learning rate α_k using

$$\alpha_k = \begin{cases} \dfrac{1}{\|g_k\|} & k=1 \\ 2*S(\alpha)-1 & otherwise \end{cases}$$

Step (5): Use backtracking line search to satisfying Wolfe conditions (3) and (4)to accept α_k

Step (6): Update the weights

$$w_{k+1} = w_k + \alpha\ d_k$$

Step (7): Set k=k+1 and go to Step (2).

3.2 Convergence Analysis

In order to establish the global convergence result, the following definition [16] is essential: the angle θ_k between the search direction d_k and steepest descent direction $-g_k$, defined by

$$\cos\theta_k = \frac{-g_k^T d_k}{\|g_k\|\|d_k\|} \tag{6}$$

Also the Zoutendijk theorem [16] necessary to establishing the convergence of any line search methods

Zoutendijk theorem [16]

Consider any iteration of the form $w_{k+1} = w_k + \alpha_k d_k$, where d_k is descent direction i.e $d_k^T g_k < 0$ and α_k satisfies the Wolfe conditions(3) and (4). Suppose that E is bounded below and continuously differentiable in an open setאcontaining the level set $\Gamma = \{w : E(w) \le E(w_1)\}$ where $w_1$starting weight vector of the iteration. The gradient vector g_k is Lipschitz continuous onא that is there exist $L > 0$ such that

$$\|g(x) - g(y)\| \le L\|x - y\| \qquad \forall\ x,\ y \in \text{א}$$

Then

$$\sum_{k=1}^{\infty} \cos^2\theta_k \|g_k\| < \infty \tag{7}$$

Now we are ready to prove the convergence of the proposed (ABBP) training algorithm as stated in the following theorem (1):

Theorem (1)

Suppose that the sequence $\{w\}_{k=1}^{\infty}$ generated by the ABBP algorithm, and assume that the assumptions of the Zoutendijk theorem are valid then

$$\lim_{k\to\infty} \|g_k\|^2 = 0$$

Proof

From second Wolfe condition

$$(g_{k+1} - g_k)^T d_k \geq (c_2 - 1)\, g_k^T d_k \tag{8}$$

While the Lipschitz condition implies that

$$(g_{k+1} - g_k)^T d_k \leq L\alpha_k \|d_k\| \tag{9}$$

By combining (8) and (9)

$$\alpha_k \geq \frac{(c_2 - 1)}{L\|d_k\|^2} g_k^T d_k \tag{10}$$

Substitute the inequality (10) in the first Wolfe condition (7)

$$E_{k+1} \leq E_k - c_1 \frac{(1-c_2)}{L} * \frac{(g_k^T d_k)^2}{\|d_k\|^2}$$

$$\rightarrow \quad E_{k+1} \leq E_k - \delta \cos^2 \theta_k \|g_k\|^2$$

Where $\delta = c_1 \dfrac{(c_2 - 1)}{L}$. By summing the above expression over all the indices less than or equal to k we obtain

$$\rightarrow \quad E_{k+1} \leq E_1 - \delta \sum_{j=1}^{k} \cos^2 \theta_j \|g_j\|^2 \tag{11}$$

Since E is bounded below we have $E_1 - E_{k+1}$ some positive constant, for all k. Hence by taking limits for (11)

$$\rightarrow \quad \sum_{k=1}^{\infty} \cos^2 \theta_k \|g_k\|^2 < \infty \tag{12}$$

Since $d_k = -g_k$ then $\cos^2 \theta_k = 1$, it follows from (12)

$$\therefore \quad \lim_{k\to\infty} \|g_k\|^2 = 0$$

4. EXPERIMENTAL RESULTS

In the following section we shall present experimental results in order to evaluate the performance of the proposed (ABBP) algorithm in three famous test problems. The problems have which been tested are the continuous function approximation problem, symmetry problem and spect heart problem. On each problem, four algorithms have been simulated. These algorithms are the standard back propagation (CBP), the momentum backpropa-gation (MBP) [8], the adaptive back propagation (ABP) [20] and the (ABBP). For the simulations an HP PC compatible with Matlab Version 6.3 has been used. We have utilized Matlab Neural Network.

Toolbox Version 3.0 for the CBP, MBP and ABP algorithms. For the heuristic parameters of the previous three algorithms, Toolbox default values are used, unless stated otherwise.

The ABBP algorithm has been implemented in Matlab environment and the values of parameters c_1, c_2 and ε have been fixed to 10^{-4},, 0.5 and 10^{-5}, respectively. At the start of each simulation, the weights of the network have been initialized by the Nguyen - Widrow method [15]. Each algorithm has been tested for the same initial weights. During training of the network, each time step is called an epoch and is defined to be a single sweep through all training patterns. At the end of each epoch, the weights of the network have been updated.

The reported parameters are min the minimum number of epochs, mean the mean value of epochs, Max the maximum number of epochs, Tav the average of total time and Succ, the succeeded simulations out of (50) trails within error function evaluations limit.

If an algorithm fails to converge within the above limit considered that it fails to train the FFNN, but its epoch are not included in the statically analysis of the algorithm, one gradient and one error function evaluations are necessary at each epoch.

1. The Continuous Function Approximation Problem

The first problem we have been considered is the approximation of the continuous function F(x)= (3x-1)/(5x+2)on the closed interval [0,2]. This problem maps one real input to a single real output. The input values are 20 equally spaced points $x_i \in$ [0, 2] and the target values are the mapping of these points from function F (x). As it is cleared, we have 20 patterns and each pattern is consisted of one input x ∈ [0, 2] and one target value F (x). The selected architecture of the FNN is 1-10-1 with logistic neurons with biases in the hidden layer and with a linear output neuron with bias. The error goal has been set to 0.001 and the maximum epochs to 2000. The results of the simulations are presented in Table 1.

Table 1: The Cont. Function Approx. Problem

Algorithms	Min	Max	Mean	Tav	Succ
CBP	fail	-------	------	------	0%
ABP	84	266	166.58	2.702	100%
MBP	692	1976	1439.7	117.50	43.33%
ABBP	68	127	59.43	2.041	100%

2. Symmetry Problem

Consider a 4-input pattern and 1-output problem [13] where the output is required to be 'one' if the input pattern configuration is symmetrical and 'zero' otherwise The selected architecture of the FFNN is 16-6-1 with logistic neurons with biases in the hidden output layer. The error goal has been set to 0.01 and the maximum epochs 2000.

Table 2: Results of Simulations for the Symmetry Problem

Algorithms	Min	Max	Mean	Tav	Succ
CBP	fail	-----	-----	------	------
ABP	117	653	224.97	1.863	100%
MBP	296	1502	1160.2	6.479	76.1%
ABBP	101	182	89.667	1.305	100%

Problem (1): (SPECT Heart Problem)

This data set contains data instances derived from Cardiac Single Proton Emission Computed Tomography (SPECT) images from the University of Colorado [14]. The network architectures for this medical classification problem consists of one hidden layer with 6 neurons and an output layer of one neuron. The termination criterion is set to $E_{rr} \leq 0.1$ within limit of 2000 epochs, table 3 summarizes the result of all algorithms.

Table 3: Results of Simulations for the Heart Problem

Algorithms	Min	Max	Mean	Tav/s	Succ
CBP	---	----	---	----	0%
ABP	98	497	321.8	0.927	98.65%
MBP	114	444	262.12	1.63	67.34%
ABBP	81	300	274.3	0.88	100%

5. CONCLUSIONS

In this work we evaluate the performance of for algorithms for training FFNN's which are uses different methods for computing the learning rate. From the rigorous analysis of the simulation results is strongly demonstrated that the computing learning rate in the proposed method with Wolfe conditions, proving more stable, efficient and reliable learning.

REFERENCES

1. Abbo K.and Marwa J.(2012), 'Learning rate computation based on derivative of sigmoid function' Event's of the 6th Scientific Annual conference (2th international vol. 1, N(1) Science and Mathematics. College of Basic education.
2. Abbo K. and Hind M .(2013). 'Improving the learning rate of the backpropagation by Aitken process'. Iraqi J. of Statistical Sciences(23).
3. Christos S. and Dimitrios S. (1996), 'Neural Networks'. Computer Science Deptt. University of U.K. Journal, Vol. 4.
4. Daniel S., Vladimir K. and pospichal J. (1997). "Introduction to multi-layer feed-forward neural networks", Chemo metrics and Intelligent laboratory system 39.
5. Dongare A., Kharde R. and Amit D. (2012), 'Introduction to Artificial Neural Network'. International Jou. of Engineering and Innovative Technology (IJEIT) Vo. 2, Issue(1)
6. Enrique C., Bertha G., Oscar F. and Amparo (2006), 'A very fast learning method for neural network Based on sensitivity, Analysis', J. of Machine learning Research7.
7. Haykin S. (2005), 'Neural Networks', Second edition by, Prentice Hall of India.

8. Jacobs J.(1988), 'Increased rates of convergence through learning rate adaptation', Neural Networks, 1.
9. Kohonen T.(1988), 'Self-organization and associative Memory', Springer Verlag, Berlin.
10. Kuan G. and Hornik K. (1991), 'Convergent of Learning algorithms with constant learning rates', IEEE Trans. Neural Networks, (2).
11. Kotsiopoulos A., Sotiropoulos D. and Grapsa T.(2004), 'A new efficient variable learning rate for Perry's spectral conjugate gradient training method'. First International Conference. From Scientific Computing to Computational Engineering.
12. Livieris I. and Pintelas P. (2011). 'An improved spectral conjugate gradient neural network training Algorithm', International J. on Artificial Intelligence Tools.
13. Ludwig O. and Nunes U. (2010), Novel Maximum-Margin Training Algorithms for Supervised Neural Networks. IEEE Transactions on Neural Networks, vol.21, issue 6.
14. Livieris I., Sotiropoulos D. and Pintelas P. (2009), 'On Descent spectral CG algorithms for Training Recurrent Neural Network, IEEE Computer Society. 13th Panhellenic, Conference on Informatics.
15. Nguyen D. and Widrow B. (1990), 'Improving the learning speed of 2-layer neural network by choosing initial values of the adaptive weights'. IEEE First,International Jaint Conference on neural Networks, (3).
16. Nocedal J and Wright S.(1999), 'Numerical Optimization. Springer-Verlag, New York.
17. Robert J. (1997), 'Artificial Neural Networks'. McGraw-Hill International Editions.
18. Rumelhart D.E, Hilton G.E. and Williams R. J(1986), 'learning representations by back-propagation errors' Nature, 323.
19. Srikanth R. (2009), 'Application of ANN in Condition Monitoring: A Case Study' (Conference proceeding 'Condition Monitoring of Mechanical System'), Gitam University, Vishakhapattanam.
20. Vogl T., Mangis R, Rigler J, Zink W. and Alkon D. (1988), Accelerating the convergence of theback-propagation method, Biological Cybernetics, 59.